文章编号:1005－0523(2004)05－0033－04

应用多 Agent 的异步处理提高汉诺塔执行效率的研究

曾 辉

(华东交通大学 信息工程学院,江西 南昌 330013)

摘要:汉诺塔是典型的递归问题,本文通过讨论多个 Agent 各自进行异步的移动,来改善传统汉诺塔的执行效率,并得出两个相应的结论.

关 键 词:Agent;多 Agent 系统;汉诺塔
中图分类号:U279.3+24 **文献标识码**:B

0 前 言

汉诺塔问题是将一堆不同大小的圆盘(Disk)通过一个辅助塔座(Peg)从一塔座移动到另一个塔座上.每次只能移动一个圆盘,而且任何时刻都不能将一个较大的圆盘压在较小的圆盘上.图 1 显示了三个圆盘问题的初始状态和目标状态.

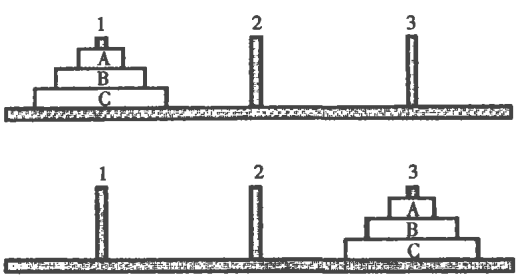

图 1 汉诺塔的初始和目标状态

该问题早已进行了广泛的研究,其中主要的研究都集中在提取不同的搜索路径中进行提取,但大多数都忽略了在提取中问题的关键所在一状态空间,而从 Agent 的观点提高解决状态空间来提高"汉诺塔"的运行效率,在文献上主要来有两个:一为卡耐基美伦大学的 Craig A. Knoblock[1](1994)与密西根大学的 Montgomery & Durfee(1999).前者证明汉诺塔的算法的复杂度可以由 $O(2^N)$ 改善为 O(n),其中 n 表示 Disk(圆盘)的数量;后者则证明可以由 O(n) 改善为 $O(\log_k n)$, 其中 k 表示将 n 个 Disk 分成 k 个均等份.显然地,来自两所大学的文章皆未使用多 Agent 方法,因此,本文将分析并应用多 Agent 异步式并行处理来分析提高汉诺塔的执行效率.

1 Agent 与多 Agent 系统的定义

Agent 是指在某一环境下具有自主性、交互性、主动性、社会性等待征的计算实体.自主性:Agent 具有属于其自身的计算资源和执行自身行为的机制,能根据外界环境的变化,由自身的内部状态和感受到的外界环境,决定自身的行为,这种转换不需要外界环境的干预,是由 Agent 自身完成的.交互性:Agent 能够与其他的 Agent 进行多种形式的交互,能够与其他 Agent 进行协调,以达到合作求解的目的.反应性:Agent 应该可以主动感知外界环境的变化,由此作出相应的对自身行为的调整.主动性:在社会性的意义下,Agent 可以根据其承诺、义务等社会属性,表现出对目标的主动完成.

目前尚没有关于 Agent 的严格定义,Wooldrige

收稿日期:2004－06－15
作者简介:曾辉(1973－),男,江西赣州人,华东交通大学讲师.

在 Intel1igent Agents 一文中给出了以下几种定义：(1)自主能力(autonomy)·Agent 可以在没有人或其他 Agent 直接干预的情况下运作，而且对自己的行为和内部状态有某种控制能力·(2) 社会能力(sociability)·Agent 和其他 Agent 通过某种交流语言进行交互·(3)反应能力·Agent 观察其环境(可能是物理世界、图形世界、或者其他 Agent)的变化，并在一定时间作出反应，以改变其环境·(4)预动能力(preactiveness)·Agent 不仅能够简单地对环境作出反应，而且能够通过接受某些启示信息，体现出一种面向目标的主动行为·

至于多 Agent 系统(Multi－Agent System；MAS)系指多种 Agent 所形成的分布式环境，在多 Agent 系统环境中各种类型的 Agent 可协同其个别技术、知识、目标和计划来解决分布式问题，完成整体多 Agent 的目标·

2 从 Agent 的观点提高汉诺塔的执行效率

2.1 传统汉诺塔的状态空间、移动过程与程序实现

1) 状态空间

在单机运作模式下，若是要完成汉诺塔中所有 Disk 的移动，就必须一步一步依序将 Pegs 搬动·下面以三个 Pegs 为例，在图 2 中显示出状态空间为 27 个·而图 3 将状态空间加以规范为 3 类型，由于 DiskC 只有二种可能性，即由 Peg1 移至 Peg2 或 Peg3·其中 Agent3 负责解决 DiskC，相同的道理 Agent1 与 Agent2 各负责解决 DiskA 与 DiskB·Agent3 扮演了关键性的角色，因为一旦它完成了之后，汉诺塔已然是完成了一大半·

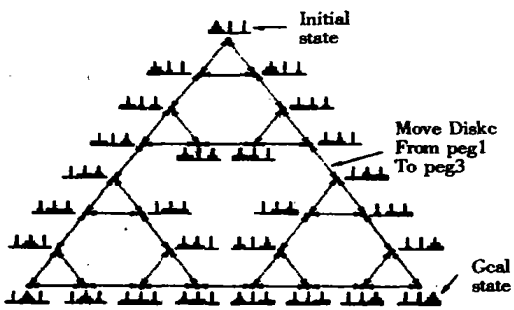

图 2 搜寻空间为 27 个

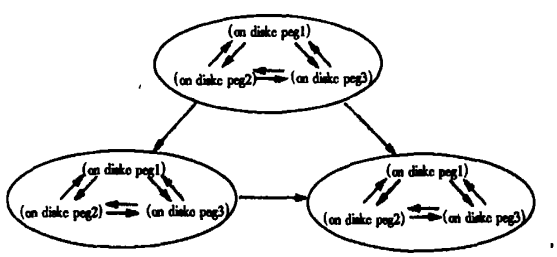

图 3 将搜寻空间加以规范为 3 类型

2) 移动过程

设汉诺塔的初始状态如图 1 所示，Disk 由小至大依序命名为 DiskA、DiskB、DiskC，三个 Peg(塔座)命名为 Peg1，Peg2，Peg3，则搬动的顺序如表 1 所示·

表 1 三个 Disk 的汉诺塔的移动过程

步骤	搬动过程
1	将 DiskA 从 Peg1 搬到 Peg3
2	将 DiskB 从 Peg1 搬到 Peg2
3	将 DiskA 从 Peg3 搬到 Peg2
4	将 DiskC 从 Peg1 搬到 Peg3
5	将 DiskA 从 Peg2 搬到 Peg1
6	将 DiskB 从 Peg2 搬到 Peg3
7	将 DiskA 从 Peg1 搬到 Peg3

想要完成整个工作必须要有七次的移动，亦即也就需要七个移动的时间才能够完成整个过程，如图 4 所示·

3) 汉诺塔的 C 程序代码实现要利用递归处理来完成·C 程序如下：

```
Void hanoi(int n, char x, char y, char z)
{  if(n==1) move(x,1,z);
else{
hanoi(n－1,x,z,y);
move(x,n,z);
hanoi(n－1,y,x,z);
}
}
```

2.2 多 Agent 分布式移动

1) 两个 Agent 的运作考虑，如果利用 Agent 的分布式并行处理能力，假设有两个 Agent 同时在运作，分别为 Agent1、Agent2，并将移动的工作分割成两个部分来运作，则处理的步骤如图 5 所示，Disk 移动的过程如表 2 所示·

表 2 双 Agent 下三个 Disk 的汉诺塔的移动过程

步骤	搬动过程
1	将 DiskA 从 Peg1 搬到 Peg3，将 DiskB 从 Peg1 搬到 Peg2
2	将 DiskA 从 Peg3 搬到 Peg2，将 DiskC 从 Peg1 搬到 Peg3
3	将 DiskA 从 Peg2 搬到 Peg1，将 DiskB 从 Peg2 搬到 Peg3
4	将 DiskA 从 Peg1 搬到 Peg3

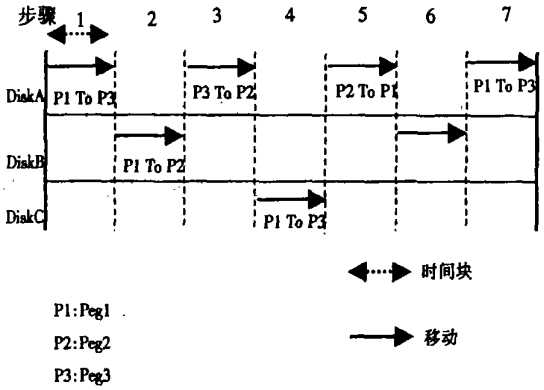

图 4　汉诺塔的移动过程

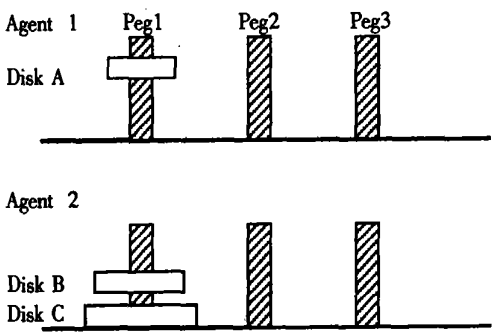

图 5　Agent1 负责移动 DiskA,Agent2 负责移动 DiskB 和 DiskC

则其移动的流程如图 6 所示:

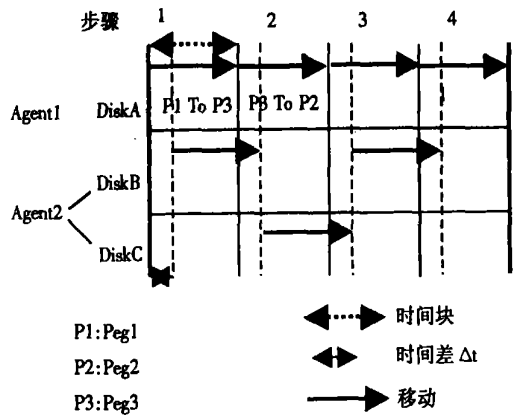

图 6　两个 Agent 同步运作

在图 6 中,虽然在移动过程中或许会出现时间差△t,但是只要前者动作一产生,后者就可以跟着产生动作,所以时间差将会非常的小,因此可以忽略不计.这与原本单机运作的模式,必须等到一个步骤完成才可以做下一个动作,在时间效率上还是有所差别的.因此可以说,利用多 Agent 的处理模式,可以将原本需要七个移动时间的动作缩减到只需要四个移动时间.

2) Agent 数量的取舍

在图 6 汉诺塔的范例中,由于放置 Disk 的 Peg 共有三个,因此在同一时间区块内,最多只能够有两个 Disk 进行移动,也就是同时只有两个 Agent 在进行工作的处理,如果有三个 Agent,则它大多数的时间将处于等待状态.因此,若使用超过三个 Agent 来处理本问题,则第三个以后的 Agent 将会一直处于等待的状态,直到前两个 Agent 完成工作,这样一来,对提高效率并没有多大的帮助.因此图 6 汉诺塔的范例中,只需要使用两个 Agent 即可完成处理工作.但是若在其它情况之下(如 Disk 数量比较的情况下),若能够在将工作细分成许多小工作,则 Agent 的数量将会与工作的效率成正比关系.

3) 双 Agent 下三个 Disk 以上的情形

在四个 Disk 等待移动的情况下,就形成了一种类似递归的处理方式.只要先将三个 Disk 的目标塔与辅助塔交换,然后在三 Disk 搬移动作的最后一个步骤(DiskA 的搬移),同时将第四个 Disk 从 Peg^1 移到 Peg^3,接下来的动作就再重复一次三个 Disk 的搬移动作,即可完成四个 Disk 的汉诺塔搬移.因此四个 Disk 的搬移就只需要 $4\times2=8$ 个时间块,即可完成.

依此类推,五个 Peg 将只需要四个 Peg 的两倍时间量,也就是 $8\times2=16$ 时间量.将传统移动次数与多 Agent 移动次数作一比较,如表 3 所示.

表 3　传统移动与 Agent 移动时间比较

Pegs 数量	传统递归处理	双 Agent 处理
1	1	1
2	3	2
3	7	4
4	15	6
…		
N	2^N-1	2^{N-1}

4) 由三个 Agent 的运作考虑

在图 7 中则是将工作指派给三个不同的 Agent,最上层是由 $Agent^3$ 负责移动 DiskC,一旦 $Agent^3$ 完成时,$Agent^2$ 只剩下一步骤(即由 peg^2 搬至 peg^3),$Agent^1$ 只剩下二步骤(即由 peg^2 搬至 peg^1 与由 peg^1 搬至 peg^3).因此,若 b 表 peg 的数量,而 n 表 Disk 的数量,则移动次数自然是由传统的 $O(b^n)$ 改善为 $O(n)$.由图 7 中可知 $Agent^3$ 进行移动 DiskC 之前,Agent1 与 $Agent^2$ 已经各自移动它们所应负担的工作.由图 8 可得知,事实上 DiskC 的行走路径只有二条路可走,即由 peg^1 至 peg^2 或 peg^1 至 peg^3,如此一来,大大的降低了汉诺塔的状态空间.

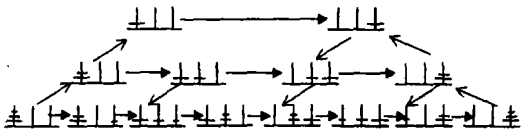

图 7 最上层是由 Agent1, Agent2 和 Agent3 负责移动 DiskA, B, 和 C

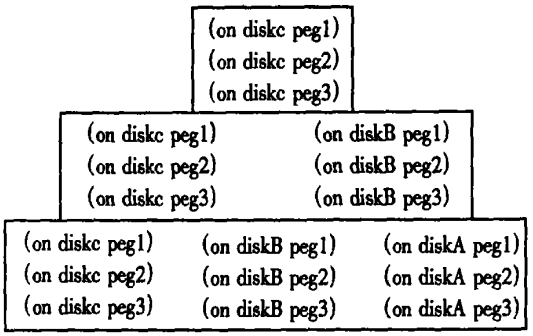

图 8 搜寻空间的处理, Agents3 扮演了关键性的角色 .

3 结 论

通过本文的分析可以得出, 由 Agent 的各自独立的分布式理念正可以用来提高汉诺塔的执行效率, 即可有效的降低资源的浪费. 并得到两个结论, 即 1. 在双 Agent 的情况下, 能够将原本执行汉诺塔所需要的时间由 $O(2^N)$ 降低成为 $O(2^{N-1})$, 这已经表示多 Agent 系统在提高工作效能上已达到降低一半的时间; 2. 多 Agent 下, 其算法更进一步发展成为由 $O(2^n)$ 改善为 $O(n)$. 因此, 若能有效利用多 Agent 理念, 充分发挥资源的使用率, 必能提高系统的效率.

参考文献:

[1] Craig A. Knoblock, Abstracting the Tower of Hanoi[EB/OL], http://www.isi.edu/info — agents/papers/knoblock90 — hanoi.pdf, 1994.

[2] Durfee. E. H., Distributed Problem Solving And Planning[M]. The MIT press, 1999.

[3] Nwana. H. S, Software Agents: An Overview. Knowledge Engineering Review[C]. 1996, 11, (3)3.

[4] Bellifemine, F., Caire, G., Trucco, T., JADE Programmer's Guide[A], 2000.

[5] 张新良, 石纯一. 多 Agent 合作求解[J]. 计算机科学, 2003.

[6] 杨锦潭, 张彦宇. 应用代理人的异步处理 提升算法效率之探讨[C]. 第 17th 科教年会论文集. 2001.

[7] 严蔚敏, 吴伟民. 数据结构(C 语言版)[M]. 北京: 清华大学出版社, 2003.

Research on Employing the Multi—Agent's Asynchronous Processing to Enhance Tower of Hanoi's Operating Efficiency

ZENG Hui

(School of Information Engineering, East China Jiaotong University, Nanchang 330013, China)

Abstract: Tower of Hanoi is a typical problem of recursion. This article aims to improve the performing efficiency of traditional Tower of Hanoi and get two corresponding conclusions by means of discussing multi—agents' respective asynchronous movement.

Key words: agent; multi agent system; tower of hanoi